LIST OF CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application.

1. (Currently amended) A computing system incorporating ~~a~~ data structures enabling rapid insertion of data segments, comprising:

a data structure including,

a head functioning as a first pointer to a first leaf node of a sorted tree ~~the first data~~ structure;

a tail functioning as a second pointer to a second leaf node of the sorted tree ~~the first data~~ structure; ~~and~~

a body, physically adjacent to the head and to the tail, having a set of pointers pointing to contiguous empty nodes of the sorted tree ~~first data~~ structure, sorted tree ~~first data~~ structure, the body including empty space distributed among non-empty leaf nodes; ~~and~~ wherein the ~~a redistribution~~ data structure is configured to examine the first leaf node to copy a leaf node index of the head to a leaf node associated with the tail, wherein indices within the first data structure are updated according to an order based on a direction the ~~redistribution~~ data structure is traversing the sorted tree ~~the first data~~ structure;

wherein the ~~first~~ data structure is configured to transfer contents of the head to the tail and delete the contents of the head when inserting a data segment such that the empty space is evenly distributed among non-empty leaf nodes.

Claims 2-3 (Cancelled)

4. (previously presented) The computing system of claim 1, wherein the nodes of the tree structure are indexed.

5. (previously presented) The computing system of claim 1, wherein each of the first and second leaf node nodes comprises a number of data segments.

6. (Currently amended) A computer implemented method containing ~~for rapid~~ insertion of data segments into a sorted tree ~~data~~ structure, comprising:

providing a data structure separate from the sorted tree structure, the data structure having;

~~preparing a redistribution data structure separate from the sorted tree structure, said redistribution data structure having~~

a head portion representing a first pointer to a first leaf node of the sorted tree structure,

a tail portion representing a second pointer to a second leaf node of the sorted tree structure, and

a body, logically adjacent to the head portion and to the tail portion, having a set of pointers pointing to contiguous empty nodes of the sorted tree structure, the body including empty space distributed among non-empty leaf nodes;

the ~~preparing~~ providing of the ~~redistribution~~ data structure including,

examining a leaf node of the tree structure associated with the head portion of the data structure;

identifying a leaf node index associated with the head portion;

copying the leaf node index to a leaf node of the tree structure associated with the tail portion of the data structure;

determining a direction in which the data structure is traversing the sorted tree structure;

updating indices within the sorted tree structure according to an order based upon the direction in which the data structure is traversing the sorted tree structure;

copying contents of the head portion to the tail portion; ~~and~~

deleting contents of the head portion of the data structure when inserting a data segment into the sorted tree structure such that empty space is evenly distributed among non-empty leaf nodes; and

~~a~~ redistributing ~~of~~ the contiguous empty ~~tree~~ nodes of the sorted tree structure by employing the ~~redistribution~~ data structure, ~~which enables to enable a more rapid insertion of the data segments,~~

wherein at least two contiguous empty nodes are maintained for the life of the data structure.

7. (previously presented) The method of claim 6, wherein the data segments are ~~may be~~ ~~is~~ inserted in any order.

8. (Previously presented) The method of claim 6, wherein the sorted tree structure comprises non-leaf and leaf nodes.

9. (previously presented) The method of claim 6, wherein the ~~tree~~ nodes of the sorted tree structure are indexed.

10. (Previously presented) The method of claim 6, wherein each of the first and second leaf nodes comprises a number of data segments.

11. (Canceled)

12. (currently amended) The method of claim 6, wherein the step of redistributing the contiguous empty nodes redistribution process ~~comprises the data structure~~ includes traversing the sorted tree structure in one of a first direction and a second direction.

13. (currently amended) The method of claim 12, wherein the first direction ~~comprises~~ is represented by a logical one and the second direction ~~comprises~~ is represented by a logical zero.

Claims 14-45 (cancelled)

46. (previously presented) The method of claim 6, wherein the sorted tree structure ~~may be is reverse~~ is sorted from a bottom of the sorted tree structure to a top of the sorted tree structure.

47. (Cancelled)

48. (previously presented) The method of claim 6, wherein the ~~redistribution process step of~~ redistributing the contiguous empty nodes maintains a consistent lookup operation on the sorted tree structure.

49. (original) The method of claim 6, further comprising:

moving the tail portion of the data structure according to a pre-calculated increment.

50. (original) The method of claim 6, further comprising:

re-distributing empty leaf nodes within the tree structure among non-empty leaf nodes of the data structure.

51. (original) The method of claim 6, wherein if the direction in which the data structure is traversing the tree structure is towards a beginning of the tree, then the method includes,

updating the indices between the tail portion and a nearest non-empty leaf node from a lowest tree level to a highest tree level.

52. (original) The method of claim 51, further comprising:

updating a remainder of the tree structure from the highest tree level to the lowest tree level.

53. (original) The method of claim 6, wherein if the direction in which the data structure is traversing the tree structure is towards an end of the tree, then the method includes,

updating the indices from a lowest tree level to a highest tree level.